DISKREETTI LAGRANGEN MENETELMÄ LAUSELOGIIKAN TOTEUTUVUUDEN TARKASTUKSEEN

Seminaariesitelmä Tietojenkäsittelytieteen seminaariin keväällä 2003

Pasi Virtanen

Alkuperäinen menetelmä: Yi Shang ja Benjamin Wah

1.Johdanto

Yi Shang ja Benjamin Wah esittivät kesäkuussa 1996 *Journal of Global Optimisation* lehdessä diskreettiin Lagrangen kerroinmenetelmään perustuvan menetelmän lauselogiikan toteutuvuustarkistukselle, eli menetelmä pyrkii toisin sanoen ratkaisemaan niin sanotun SAT-tehtävän. Menetelmä on epätäydellinen, eli se ei pysty tunnistamaan toteutumatonta lausejoukkoa, mutta se on kuitenkin useita kertaluokkia täydellisiä menetelmiä nopeampi menetelmä ja julkaisun aikaan myös muita kilpailevia epätäydellisiä menetelmiä parempi. Menetelmässä toteutuvuustarkastus muunnetaan kokonaislukuoptimointitehtäväksi ja sen ratkaisu yhdistää toteutumattomien klausuulien määrän minimointiin perustuvan lokaalin haun lagrangen kertoimiin jotka rankaisevat menetelmää toteutumattomista klausuuleista ja nostavat tämän kautta menetelmän pois lokaaleista minimeistä. Lagrangen kertoimien vuoksi menetelmä ei vaadi satunnaisia uudelleen aloituksia päästäkseen pois lokaaleista minimeistä. Menetelmän hyviin puoliin kuuluu se, että siinä on suhteellisen vähän parametreja joita käyttäjän tarvitsee säätää, vaikkakin vaikeille ongelmille menetelmää kannattaakin modifoida.

Käsittelyä varten SAT-tehtävä formuloidaan seuraavasti. Olkoon tehtävä konjunktiivisessa normaalimuodossa, jossa on yhteensä n subjunkteista muodostuvaa klausuulia $\{C_1, C_2, ..., C_n\}$ ja m muuttujaa $x = (x_1, x_2, ..., x_m)$, $x_i \in \{0,1\}$. Tehtävänä on etsiä x:lle arvot siten, että jokainen klausuuleista $\{C_1, C_2, ..., C_n\}$ toteutuu eli toisin sanoen ainakin yksi klausuulissa esiintyvistä ehdoista on totta. Tässä menetelmässä tehtävä halutaan muuntaa diskreetiksi optimointitehtäväksi seuraavaan tapaan. Tehtävänä on minimoida N(X).

$$\min_{x_i \in \{0,1\}} m \; N(x) = \sum_{i=1}^{n} U_i(x) \qquad (1)$$

ehdoilla $U_i(x) = 0 \;\; \forall i \in \{1,2,...,n\}$,

jossa U_i=0 jos C_i on toteutunut ja U_i=1 jos C_i ei ole toteutunut. Tähän tehtävään voidaan soveltaa Lagrangen kertoimiin perustuvaa menetelmää.

Lagrangen kertoimien menetelmää on perinteisesti käytetty rajoitettujen jatkuvien optimointiongelmien ratkaisuun, joten menetelmän soveltamiseksi tähän tapaukseen Shang ja Wah ovat esittäneet diskreetin lagrangen menetelmän (DLM), jotta vastaavaa menetelmää kuin jatkuville muuttujille voidaan käyttää nyt kun muuttujat ovat diskreettejä. Myöhemmin näytetään, että tämä menetelmä löytää SAT-tehtävän tapauksessa tehtävälle globaalin optimin todennäköisyydellä 1, mutta ei kuitenkaan välttämättä äärellisessä ajassa, menetelmä ei siis voi taata ratkaisun löytymistä

vaikka se olisi olemassa. Jos annetulla tehtävällä ei ole ollenkaan ratkaisua DLM ei pysähdy koskaan.

2. Muista SAT-tehtävien ratkaisumenetelmistä

Shang ja Wah esittelevät paperissaan myös muita tapoja SAT-tehtävien ratkaisemiseen. Vaikka ala on niin nopeasti kehittyvä, että tuolloisella tietämyksellä kirjoitetut näkemykset ovatkin osittain vanhentuneita niin käydään silti läpi SAT-ratkaisualgoritmien kenttää niin kuin se nähtiin artikkelin julkaisun aikaan, jotta lukija osaa sijoittaa esitetyn menetelmän oikeaan kontekstiin.

2.1 Diskreetit menetelmät

Diskreetit menetelmät voidaan jakaa rajoitettuihin, rajoittamattomiin ja täydellisiin ja epätäydellisiin menetelmiin.

(a) Diskreetit ja rajoitetut toteutuvuusmenetelmät

Diskreetit ja rajoitetut toteutuvuusmenetelmät perustuvat tehtävän esittämiseen logiikan syntaksia hyväksi käyttäen. Nämä menetelmät ovat diskreettejä, koska muuttujat saavat vain arvoja tosi ja epätosi. Rajoitettuja menetelmistä tekee se, että annetussa yhtälössä annetaan rajoituksena se, että kaikkien klausuulien on toteuduttava. Täten mikä tahansa käypä ratkaisu yhtälölle on myös optimaalinen. Nämä menetelmät voivat olla joko täydellisiä tai epätäydellisiä riippuen siitä pystytäänkö niillä toteamaan lausejoukko toteutumattomaksi vai ei. Tähän luokkaan kuuluvat esimerkiksi Davis-Putnam-Loveland-Logemann algoritmiin perustuvat menetelmät, jotka ovat tällä hetkellä suosituimpia SAT-tehtävän täydellisiä ratkaisualgoritmeja. Näiden menetelmien haitta on se, että ne ovat laskennallisesti vaativia eivätkä täten pysty ratkaisemaan suuria ongelmia joissa on paljon literaaleja

(b) Diskreetit ja rajoittamattomat menetelmät

Tämän tyyppisissä menetelmissä lähdetään minimoimaan toteutumattomien klausuulien määrää, eli optimointitehtävä on

$$\min_{x \in \{0,1\}^m} N(x) = \sum_{i=1}^{n} U_i(x)$$

Jossa $U_i(x)$ on 0 jos annettu totuusjakelu x toteuttaa C_i:n ja 1 muuten. Tässä tapauksessa jos SAT-tehtävälle löytyy ratkaisu niin N:n arvoksi tulee 0. Tämän tyyppisiä lokaaleja hakumenetelmiä on esitetty runsaasti. Lokaali hakumenetelmä tarkoittaa tässä tapauksessa sitä, että algoritmi ei välttämättä käy läpi koko hakuavaruutta. Lokaalien hakumenetelmien haitta on se, että ne jäävät usein jumiin lokaaleihin minimeihin, eli pisteisiin joiden ympäristöstä ei löydy parempaa pistettä. Tällöin käytetään usein uudelleenaloitusta satunnaisesta paikasta, mutta se ei takaa sitä, että menetelmä ei tälläkin kerralla juuttuisi lokaaliin minimiin. Useimmat tämän tyyppisistä menetelmistä ovat epätäydellisiä, mutta esimerkiksi backtrackingin avulla ne voivat olla täydellisiäkin. Esimerkkinä tämän kaltaisesta menetelmästä Shang ja Wah antavat Selmanin kehittämän GSATin jota käytetään myöhemmin vertailukohteena DLM:lle. GSAT aloittaa haun satunnaisesta paikasta ja etenee yksittäisiä totuusarvoja vaihtamalla kunnes ratkaisu löytyy tai säädeltävä vaihtojen suurin määrä ylittyy. Lokaaleista minimeistä GSAT pääsee eteenpäin joko vaihtamalla uuteen satunnaiseen paikkaan tai siirtymällä ylöspäin, eli vaihtamalla totuusarvoa siten, että toteutumattomien klausuulien määrä kasvaa. Tasangoille eli alueille ratkaisuavaruuden topologiassa jossa yhtä hyviä ratkaisuja on lähekkäin paljon juuttumista GSAT estää tekemällä sivuttaissiirtoja. GSAT on ahne menetelmä, eli se vaihtaa aina sen muuttujan totuusarvoa jolla saavutetaan suurin pienennys toteutumattomien klausuulien määrään.

(c) Diskreetit rajoitetut menetelmät

Nämä menetelmät perustuvat tehtävän esittämiseen yllä olevan yhtälön (1) kaltaisesti. Yksi tapa ratkaista SAT-tehtäviä on muotoilla täten tehtävä lineaariseksi kokonaislukuoptimointitehtäväksi ja ratkaista se tunnetuin menetelmin, mutta nämä menetelmät ovat laskennallisesti kalliita. Etu kohdan b-menetelmiin on se, että ehdot jotka vaativat, että kaikki klausuulit toteutetaan eivät anna algoritmin jäädä paikallisiin minimeihin. Shangin ja Wahin esittämä DLM-algoritmi kuuluu myös tähän joukkoon.

2.2 Jatkuvat menetelmät

Näissä menetelmissä tehtävä muunnetaan diskreetistä avaruudesta jatkuvaan. Nämä menetelmät ovat yleisesti ottaen kalliita eivätkä täten kelpaa kuin pienille ongelmille. Shang ja Wah esittävät

tästä kategoriasta kaksi mallia esimerkkeinä, mutta sivuutetaan ne kuitenkin liian epäkäytännöllisinä.

3. Diskreetti Lagrangen menetelmä SAT-tehtävien ratkaisussa

Tässä luvussa kuvataan miten Lagrangen kertoimien menetelmä laajennetaan SAT-tehtäviä varten jatkuvista ja rajoitetuista optimointitehtävistä diskreetteihin. Muistutettakoon vielä, että yllä olevalla tavalla muotoillulle SAT-tehtävälle kaikki lokaalit optimit ovat myös globaaleja joten menetelmän tarvitsee vain löytää lokaali minimi. Käydään ensiksi läpi Lagrangen kertoimien menetelmän yleistä teoriaa ja laajennetaan se sitten diskreetille alueelle. Tämän osan lopuksi näytetään miten Shang ja Wah käyttivät Lagrangen menetelmää SAT-tehtävien ratkaisuun.

3.1 Lagrangen kertoimien menetelmä jatkuvamuuttujaisten optimointongelmien ratkaisussa

Lagrangen menetelmiä on perinteisesti käytetty jatkuvien, rajoitettujen optimointitehtävien ratkaisuun. Menetelmät muuntavat rajoitetun tehtävän rajoittamattomaksi seuraavasti. Olkoon ratkaistavana oleva optimointitehtävä määritelty seuraavasti.

$$\begin{aligned} &\min_{x \in E^m} \ f(x) \\ &\textit{ehdolla} \quad g(x) = 0 \end{aligned} \tag{8}$$

jossa $x = (x_1, x_2, ..., x_n)$ ja $g(x) = (g_1(x), g_2(x), ..., g_n(x))$ ovat tehtävän rajoitukset. Tehtävän Lagrangen funktio F määritellään seuraavasti

$$F(x, \lambda) = f(x) + \sum_{i=1}^{n} \lambda_i g_i(x)$$

, jossa $\lambda = (\lambda_1, \lambda_2, ..., \lambda_n)$ ovat Lagrangen kertoimet. Lagrangen kertoimilla siis muutetaan rajoitettu optimointitehtävä rajoittamattomaksi. Seuraavaksi määritellään Lagrangen funktiolle satulapiste ja todistetaan, että alkuperäisen tehtävän lokaali minimi löytyy tästä satulapisteestä.

Satulapiste (x^*, λ^*) Lagrangen funktiolle $F(x, \lambda)$ on piste joka täyttää seuraavan ehdon.

$$F(x^*, \lambda) \leq F(x^*, \lambda^*) \leq F(x, \lambda^*)$$

kaikille $(x^*, \lambda\ )$ ja (x, λ^*) jotka ovat riittävän lähellä pistettä (x^*, λ^*).

Satulapiste on siis funktion $F(x, \lambda)$ paikallinen minimi x-avaruudessa ja paikallinen maksimi λ-avaruudessa. Luonnollinen tapa löytää satulapiste on siis kasvattaa funktion arvoa λ-avaruudessa ja laskea sitä x-avaruudessa. Luonnollinen tulkinta Lagrangen kertoimille on se, että ne rankaisevat kasvaessaan toteutumattomista klausuuleista ja osoittautuukin, että kun tätä rangaistusta pyritään maksimoimaan satulapistettä etsiessä Lagrangen kertoimet pakottavat rajoitukset voimaan. Todistetaan seuraavaksi Satulapisteteoreema joka todistaa, että satulapiste on todellakin alkuperäisen optimointitehtävän ratkaisu.

Satulapisteteoreema. x^* on alkuperäisen ongelman (8) lokaali minimi jos on olemassa λ^* siten, että (x^*, λ^*) Lagrangen funktion $F(x, \lambda)$ satulapiste.

Todistus: Koska (x^*, λ^*) on satulapiste, $F(x^*, \lambda) \le F(x^*, \lambda^*)$, kun λ on lähellä λ^*:ä. Lagrangen funktion määritelmästä nähdään, että tästä seuraa

$$\sum_{i=1}^{n} \lambda_i g_i(x^*) \le \sum_{i=1}^{n} \lambda_i^* g_i(x^*)$$

Oletetaan nyt, että on olemassa jokin k, $1 \le k \le n, g_k(x^*) \neq 0$. Jos $g_k(x^*) > 0$ niin vektori $\lambda = (\lambda_1^*, ..., \lambda_k^* + \delta, ..., \lambda_n^*)$ rikkoisi epäyhtälöä vastaan positiiviselle δ. Jos taas $g_k(x^*) < 0$, niin vektori $\lambda = (\lambda_1^*, ..., \lambda_k^* - \delta, ..., \lambda_n^*)$ rikkoisi epäyhtälöä vastaan positiiviselle δ. Täten täytyy olla $g_k(x^*) = 0$, eli alkuperäiset rajoitukset toteutuvat ja x^* on käypä ratkaisu alkuperäiselle ongelmalle.
Koska (x^*, λ^*) on satulapiste, $F(x^*, \lambda^*) \le F(x, \lambda^*)$, kun x on lähellä x^*:ä. Lagrangen funktion määritelmästä saadaan nyt

$$f(x^*) \le f(x) + \sum_{i=1}^{n} \lambda_i^* g_i(x).$$

Täten mille tahansa käyvälle x, g(x) = 0 ja saadaan

$$f(x^*) \leq f(x).$$

Täten x^* on lokaali minimi.

Yleensä Lagrangen kertoimiin perustuvat menetelmät etsivät satulapisteitä siirtymällä gradienttien suunnissa pienempiin arvoihin x-avaruudessa ja suurempiin arvoihin λ-avaruudessa. Gradienttien määrittelystä johtuen, ne ovat molemmat nollia ainoastaan satulapisteessä joten menetelmä jatkaa toimintaansa kunnes se löytää lokaalin minimin. Nämä menetelmät palauttavat siis ainoastaan käypiä lokaalisia minimejä, mutta SAT-tehtävässähän lokaalit minimit ovat myös globaaleja minimejä eli tämän tyyppinen menetelmä löytää globaalin optimin.

3.2 Lagrangen menetelmä diskreetille ongelmalle

Diskreettien muuttujien tapauksessa ongelmana on löytää sopiva gradienttioperaattori jolla satulapiste voidaan löytää. Olkoon tehtävä määritelty samaan tapaan kun yllä, mutta nyt niin, että kaikki muuttujat ovat diskreettejä. Määritellään lokaali minimi tälle tehtävälle niin, että piste on lokaali minimi jos ei ole olemassa yhtään pistettä joka eroaa tästä pisteestä korkeintaan yhdessä x_i joka antaa kohdefunktiolle pienemmän arvon. Piste on siis lokaali minimi jos kohdefunktion arvoa ei saada pienemmäksi muuttamalla vain yhden muuttujan arvoa. Shang ja Wah määrittelevät gradientti operaattorin Δ_x seuraavasti:

$$\Delta_x F(x,\lambda) = (\delta_1, \delta_2, ..., \delta_n) \in \{-1,0,1\}^m, \sum_{i=1}^{m} |\delta_i| = 1, \; ja\left(x - \Delta_x F(x,\lambda)\right) \in \{0,1\}^m$$

, jossa lisäksi korkeintaan yksi kertoimista δ_i eroaa nollasta. Lisäksi mille tahansa vain yhdessä x_i:ssä tästä pisteestä eroavalle x:n arvolle x’ täytyy päteä

$$F(x - \Delta_x F(x,\lambda), \lambda) \leq F(x', \lambda)$$

Jos $\forall x', F(x,\lambda) \leq F(x',\lambda)$ niin $\Delta_x F(x,\lambda) = 0$

Gradienttioperaattori määrittelee siis sen muuttujan jonka arvoa muuttamalla saadaan suurin parannus kohdefunktion arvossa. Jos piste on lokaali minimi, eli kohdefunktion arvo on suurempi

kaikissa pisteissä jotka ovat Hamming-etäisyydellä 1 se saa arvon 0. Nyt voidaan määritellä yleistetty diskreetti Lagrangen menetelmä (DLM)

Yleistetty diskreetti Lagrangen menetelmä (DLM)

$$x^{k+1} = x^k - D(x^k, \lambda^k)$$

$$\lambda^{k+1} = \lambda^k + g(x^k)$$

$D(x^k, \lambda^k)$ on joku heuristinen laskeutumissuunta x:n päivittämiselle ja k on iteraatioindeksi.

$D(x^k, \lambda^k)$ ei ole suinkaan yksiselitteinen vaan mikä tahansa heuristiikka joka johtaa x:ää kohti optimia kelpaa. Kaksi Shangin ja Wahin esittämää tapaa ovat ns. steepest descent –menetelmä jossa siirrytään aina siihen suuntaan jossa kohdefunktion arvo laskee jyrkimmin, eli toisin sanoen diskreetin gradienttioperaattorin $\Delta_x F(x, \lambda)$ osoittamaan suuntaan. Toinen vaihtoehto on ns. hill-climbing menetelmä, jossa edetään ensimmäiseen pisteeseen tämänhetkisen x:n ympäristössä joka laskee kohdefunktion arvoa. Käytettiinpä kumpaa näistä menetelmistä tahansa voi tämä algoritmi päästä satulapisteeseen. Olennainen ehto suunnalle $D(x^k, \lambda^k)$ on, että se osoittaa aina suuntaan jossa kohdefunktion arvo pienenee ja, että sen arvo on 0 lokaalissa minimissä. Jälkimmäisestä funktiosta nähdään suoraan, että Lagrangen kertoimet muuttuvat aina kun tehtävälle määritellyt ehdot eivät ole voimassa, eli toisin sanoen g(x) != 0.
Tärkeä ominaisuus DLM:ssä on, että se jatkaa toimintaansa kunnes satulapiste löytyy eli toisin kuin monet muut epätäydelliset menetelmät tämän menetelmän ei tarvitse turvautua satunnaisiin uudelleenaloituksiin tai edes ylöspäin siirtymisiin jotta lokaaleista minimeistä päästäisiin eteenpäin. DLM on globaali hakumenetelmä joka näiden yhtälöiden kautta yhdistää sekä lokaalia, että globaalia hakua. x:n arvoja muuttamalla etsitään Lagrangen kertoimien määrittämässä avaruudessa lokaalia minimiä ja Lagrangen kertoimien arvoja muuttamalla kun lokaali minimi löytyy saadaan hakualueeksi koko avaruus.

3.3 Diskreetin Lagrangen menetelmän käyttäminen SAT-tehtävän ratkaisemiseen

Jotta menetelmää voidaan käyttää SAT-tehtävän ratkaisemiseen täytyy kehittää keinotekoinen kohdefunktio H(x). Kun kohdefunktio valitaa sopivasti tarkoittaa satulapisteen löytyminen sitä, että alkuperäinen tehtävä on ratkaistu. Välttämätön ehto tälle on yksinkertaisesti se, että SAT-tehtävän

ratkaisun täytyy olla funktion H(x) lokaali minimi. Tämä seuraa suoraan siitä, että ratkaisupisteessä kaikki rajoitusehdot toteutuvat ja täten kaikkien Lagrangen kertoimien kertomien lukujen arvo on 0. Tällöin tehtävän Lagrangen funktio muodostuu ainoastaan H(x):stä ja jotta algoritmi pysähtyisi on ratkaisun oltava lokaali minimi. Ratkaisupisteessä Lagrangen kertoimien arvoilla ei ole mitään merkitystä, koska g(x) = 0.
Tällaisia funktioita on luonnollisesti hyvin suuri määrä. Shang ja Wah esittävät kolme erilaista vaihtoehtoa funktiolle H(x). Yksi näistä pitää kohdefunktiona yksinkertaisesti toteutumattomia klausuuleja kun taas kaksi muuta painottavat muita pidempiä tai lyhyempiä klausuuleja. Omaan implementaatioonsa he valitsivat yksinkertaisimman tapauksen jossa kohdefunktio siis

$$H(x) = \sum_{i=1}^{n} U_i(x).$$

Tällä kohdefunktiolla diskreetiksi Lagrangen funktioksi saadaan

$$L(x) = \sum_{i=1}^{n} U_i(x) + \sum_{i=1}^{n} \lambda_i U_i(x) = \sum_{i=1}^{n} (1 + \lambda_i) U_i(x),$$

jossa $x \in \{0,1\}^m, U(x) = (U_1(x),...,U_n(x)) \in \{0,1\}^n$. Satulapiste tälle funktiolle on määritelty seuraavalla ehdolla.

$$L(x^*, \lambda) \le L(x^*, \lambda^*) \le L(x, \lambda^*)$$

kaikille λ riittävän lähellä λ^*:ä ja kaikille x joiden Hamming-etäisyys pisteeseen x^* on 1. Samalla tavalla kuin yllä voidaan todistaa satulapisteteoreeman diskreetti versio joka sanoo, että SAT-tehtävän globaali minimi löydetään vain ja ainoastaan diskreetin Lagrangen funktion satulapisteestä. Koska Lagrangen menetelmä pysähtyy ainoastaan löydettyään satulapisteen tiedetään, että menetelmä on aina pysähtyessään löytänyt alkuperäisen SAT-ongelman ratkaisevan totuusjakelun. Menetelmä ei tosin pysähdy varmasti rajallisessa ajassa vaikka tehtävällä olisikin olemassa jokin ratkaisu.

Näiden tulosten pohjalta Shang ja Wah esittävät DLM menetelmän A SAT-tehtäville.

Diskreetti Lagrangen menetelmä A SAT-tehtäville

$$x^{k+1} = x^k - \Delta_x L(x^k, \lambda^k)$$
$$\lambda^{k+1} = \lambda^k + U(x^k),$$

jossa siis Δ_x on diskreetti gradienttioperaattori joka kertoo mihin suuntaan x-avaruudessa täytyy siirtyä jotta saavutettaisiin suurin pienennys kohdefunktion arvossa. Tästä algoritmista nähdään helposti, että se konvergoi ainoastaan globaaliin optimiin, koska gradienttioperaattori määriteltiin siten, että se ohjaa ratkaisua aina pienempiin arvoihin kunnes lokaali optimi on löydetty. Alemmasta yhtälöstä nähdään, että Lagrangen kertoimet kasvavat aina kun tehtävässä on jäljellä toteutumattomia klausuuleja, eli algoritmi ei voi pysähtyä ennenkuin kaikki ehdot on toteutettu.

Esimerkki algoritmin A toiminnasta:

Olkoon tehtävässä neljä muuttujaa, $\{x_1, x_2, x_3, x_4\}$, 7 klausuulia,

$$\{(x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4) \wedge$$
$$(x_2 \vee x_3 \vee \bar{x}_4) \wedge (x_2 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_2 \vee x_3 \vee \bar{x}_4)\}$$

ja kaksi ratkaisua {(1,0,0,0),(0,0,1,1)}.
Algoritmi A toimii seuraavasti.

(1) Aluksi $x^0 = \{(1,1,1,1)\}$ ja $\lambda^0 = \{(0,0,0,0,0,0,0)\}$. Lagrangen funktion arvo on $L(x^0, \lambda^0) = 1$.

(2) Ympäristössä olevien pisteiden Lagrangen funktion arvot ovat nyt $L((1,1,1,0), \lambda^0) = L((1,1,0,1), \lambda^0) = L((1,0,1,1), \lambda^0) = L((0,1,1,1), \lambda^0) = 1$, eli kaikissa toteutumatta jää yksi klausuuli. Koska mikään näistä ei ole parempi kuin tämän hetkinen niin $\Delta_x L(x^0, \lambda^0) = 0$. Täten $x^1 = x^0$. Koska neljäs klausuuli jää toteutumatta on $\lambda^1 = \{(0,0,0,1,0,0,0)\}$. Tässä on huomattavaa, että koska klausuuli 4 jäi toteutumatta niin Lagrangen kertoimen päivitys rankaisee tulevaisuudessa tämän klausuulin toteutumattomuudesta.

(3) $L(x^1, \lambda^1) = 2$. Naapuripisteiden L:n arvot ovat $L((1,1,1,0), \lambda^1) = L((1,1,0,1), \lambda^1) = L((0,1,1,1), \lambda^1) = 1$ ja $L((1,0,1,1), \lambda^1) = 2$. Nyt voidaan valita gradienttioperaattorin arvoksi kolmesta vaihtoehdosta, joille kullekin L:n arvo on 1. Kaikissa tapauksissa saadaan $\lambda^2 = \{(0,0,0,2,0,0,0)\}$. Valitaan tässä esimerkissä suunta (1,0,0,0) joka johtaa pisteeseen (0,1,1,1).

(4) $x^2 = (0,1,1,1)$ ja $L(x^2, \lambda^2) = 1$. Naapuripisteille saadaan L:n arvoiksi $L((1,1,1,1), \lambda^2) = 3$, $L((0,0,1,1), \lambda^2) = 0$ ja $L((0,1,0,1), \lambda^2) = L((0,1,1,0), \lambda^2) = 1$. Edetään siis näistä pienimmän arvon antavaan pisteeseen. $x^3 = (0,0,1,1)$ ja $\lambda^3 = \{(0,0,0,2,0,1,0)\}$.

(5) Nyt U(x^3)=0 eli ratkaisu on löytynyt.

Pantakoon esimerkistä merkille, että Lagrangen kertoimien arvot eivät voi koskaan pienentyä ja isoissa probleemoissa ne kasvavat hyvin suuriksi varsin nopeasti.

Kun DLM:llä ratkaistaan tähän tapaan SAT-tehtäviä voidaan tämä menetelmä tulkita kahdella eri tavalla. Ensinnäkin Lagrangen kertoimet voidaan tulkita rangaistuksiksi joita algoritmi antaa klausuuleille joita se ei ole saanut toteutumaan. Tämä vetää algoritmia sellaisiin suuntiin jossa se toteuttaa sellaisia klausuuleita joita se ei ole ennen pystynyt toteuttamaan eli kertoimet vetävät algoritmia ratkaisuavaruudessa sellaisiin suuntiin missä ei ole vielä käyty.
Toinen järjestelmän tulkinta perustuu siihen, että ehdot ja kohdefunktio ovat toisistaan riippuvia ja muoto jonka diskreetti Lagrangen funktio sai näyttää siltä, että Lagrangen kertoimet olisivat eri klausuuleille annettuja painokertoimia kohdefunktiota määriteltäessä.

4. DLM:n implementaatioita SAT-tehtävien ratkaisemiseen

Vaikka jo esitetty perusalgoritmi toimiikin periaatteessa niin siinä on kuitenkin parantamisen varaa. Lagrangen menetelmän käytössä on kolme eri komponenttia joita voidaan muunnella: Lagrangen funktion derivaatan, eli etenemissuunnan x-avaruudessa, valinta, Lagrangen kertoimien päivitys ja rajoitusfunktioiden laskeminen.

4.1 Algoritmin suunnitteluun liittyviä seikkoja

DLM-menetelmän yleinen toimintatapa on se, että se suorittaa laskeutumisia x-avaruudessa ja nousuja λ-avaruudessa löytääkseen satulapisteen. Laskeutumissuunnan määrittämiseksi on olemassa, kuten yllä mainittiin, kaksi eri tapaa. Kummassakin tapauksessa käydään läpi käsiteltävän pisteen ympäristöä Hamming-säteellä 1. Ahneessa haussa käytäisiin läpi kaikki mahdolliset suunnat ja valittaisiin niistä pienin L:n arvo kuten ylläolevassa esimerkissä tehtiin. Shang ja Wah vertasivat tätä menetelmää useilla benchmark-testeillä kuitenkin strategiaan jossa siirrytään aina ensimmäiseen suuntaan jossa kohdefunktion arvo pienenee ja tulivat tulokseen, että ahne strategia

on kertaluokkia hitaampi löytämään ratkaisuja, joten heidän implementaatiossaan siirrytään aina ensimmäiseen suuntaan missä löydetään parannusta kohdefunktiolle.
Lagrangen kertoimien päivittämisestä Shang ja Wah panivat merkille, että näitä ei kannata päivittää joka kerta tai kertoimet painottavat hakua liiaksi Lagrangen kertoimien määräämään suuntaan. Heidän implementaatiossaan kertoimia päivitetään kahdessa eri tapauksessa: joko silloin kun algoritmi löytää lokaalin minimin tai silloin kun joku ennalta määrätty määrä T iteraatioita täyttyy. Kokeellisesti he määrittivät, että algoritmi toimii paremmin kun T:n arvo on hyvin suuri, voidaan käyttää jopa ääretöntä arvoa T:lle jolloin Lagrangen kertoimia päivitetään vain kun algoritmi löytää lokaalin minimin. Lisäksi Shang ja Wah ovat lisänneet kertoimien määritykseen kertoimen c joka kertoo kuinka paljon kertoimia kerrallaan määritetään. c on vektoriarvoinen ja voisi periaatteessa vaihdella eri ulottuvuuksien ja ajan mukaan. Yksinkertaisuuden vuoksi he ovat tosin käyttäneet testeissään arvoa c=1, paitsi joillekin suuremmille tehtäville joille on käytetty pienempää vakioarvoa.
Vaikeammille ongelmille on haluttu löytää lisäksi strategia joka pienentää Lagrangen kertoimia aina toisinaan etteivät ne kasva ajan myötä liian suuriksi. Tämä strategia esitetään myöhemmin.
Yksi ongelma diskreetissä tilanteessa on se, että haku saattaa juuttua tasangolle. Tasanko on alue ratkaisuavaruudessa jossa kohdefunktio saa saman arvon monissa vierekkäisissä pisteissä. Algoritmi saattaa nähdä tasangon lokaalina miniminä ja siksi Shang ja Wah ovat kehittäneet kaksi strategiaa joiden avulla tasankoja voidaan käydä läpi. Tasangoilla liikuttaessa Lagrangen kertoimia ei ole hyvä muuttaa aina koska tällöin alueen topologia muuttuu ja tasangon keskellä oleva optimi voi jäädä löytymättä. Tätä varten Shang ja Wah ovat kehittäneet flat moveksi kutsumansa strategian jossa algoritmi liikuu saman L:n arvon antavissa tiloissa päivittämättä Lagrangen kertoimia.
Toinen strategia tasankojen tutkimiseksi on niin kutsuttu tabu-lista joka pitää kirjaa minkä muuttujien arvoa on tasangolla muutettu jotta algoritmi voi välttää käymästä samoissa tiloissa useita kertoja.

4.2 Kolme eri implementaatiota yleisestä DLM-menetelmästä

Ottaen huomioon edellisen kappaleen seikat Shang ja Wah kehittivät kolme eri menetelmää. Menetelmä A1 on näistä yksinkertaisin. Siinä on erikoisuutena aiemmin esitettyyn se, että se käyttää kahta eri tapaa muutettavan muuttujan hakuun. Se joko vaihtaa muuttujien arvoja järjestyksessä kunnes löytyy parannusta tuova suunta tai sitten se vaihtaa vain niiden muuttujien arvoja jotka ovat toteutumattomissa klausuuleissa. Ensimmäinen tapa on nopea haun alkuvaiheessa, mutta kun haku edistyy ja toteutumattomia klausuuleja on vain vähän jäljellä on parempi käyttää

toista vaihtoehtoa. A1 käyttää parametria ϑ määrittämään kuinka monta klausuulia täytyy olla toteutumatta kunnes menetelmää vaihdetaan. Kaikissa kokeissa he käyttivät arvoa ϑ=10. Menetelmä A2 käyttää puolestaan toisenlaista strategiaa lokaaleja parannuksia tekevien muuttujien hakuun. Tämä menetelmä alkaa pitää listaa muuttujista joiden totuusarvoa vaihtamalla Lagrangen funktion arvoa voidaan parantaa. Tällöin x:n päivityssuunnan valitseminen on niin yksinkertaista, että valitaan tästä listasta ensimmäinen suunta. Tähän strategiaan menetelmä vaihtaa kun iteraatioiden määrä ylittää kolmasosan tehtävän muuttujien määrästä.
Versioon A3 on lisäksi lisätty aiemmin käsitellyt flat movet ja tabu-listat. Lisäksi A3:ssa skaalataan Lagrangen kertoimia tietyin ennalta valituin väliajoin. Muuten A3 on samanlainen kuin A2.

Algoritmit on esitetty liitteessä 1.

4.3 Syitä menetelmän A3 luomiseen

Kokeissaan Shang ja Wah huomasivat, että A2-versiolla DLM-menetelmästä oli vaikeuksia selvittää joitain vaikeampia testiongelmia. Tutkimalla graafisesti algoritmin toimintaa he huomasivat, että jotkut Lagrangen kertoimet kasvoivat hyvin suuriksi ja jatkoivat nopeaa kasvuaan, mutta toteutumattomien klausuulien määrä pysyi alun nopean tiputuksen jälkeen suurin piirtein vakiona. Tästä johtuen Lagrangen kertoimet aiheuttivat liian suurta heiluntaa Lagrangen funktion arvoihin ja tekivät haun heikommaksi jonka vuoksi menetelmää A2 parannettiin lisäämällä siihen yllä kuvatut 3 strategiaa. Flat movejen ja tabu-listan lisääminen pienensikin kertoimien kasvua, mutta ne kasvoivat silti vaikeissa ongelmissa rajatta jonka vuoksi otettiin lisäksi käyttöön skaalaus. Näiden parannusten jälkeen A3 pystyi suorittamaan tehokkaasti lähes kaikki käytetyt benchmark-testit.

5. Koetuloksia

Shang ja Wah testasivat DLM-menetelmää käyttäen DIMACS-arkiston SAT-benchmarkkeja. Menetelmä toteutettiin C-kielellä ja testejä tehtiin kaikille tehtäville joille ratkaisu on olemassa. Tuloksia verrattiin kirjallisuudesta löytyneisiin tuloksiin. Testejä tehtiin seuraavanlaisille tehtäville:

- Piirisynteesiongelmat (ii) (Kamath et al.)
- Piiridiagnoosiongelmat (ssa)
- Parity learning ongelmat (par)

- Keinotekoisesti luodut 3-SAT tehtävät (aim)
- Satunnaisesti generoidut SAT-tehtävät (jnh)
- Suuret, satunnaiset, ratkeavat 3-SAT tehtävät (f)
- Vaikeat graafinväritysongelmat (g)
- Hanoin tornit ongelma (hanoi) ja
- Gu'n asynchronous-circuit synthesis benchmarkit (as) ja technology mapping benchmarkit (tm)

Eri implementaatioista A1 toimi hyvin "aim"-ongelmiin, mutta ei yhtä hyvin muihin. A2 puolestaan toimi suhteellisen hyvin kaikkiin benchmarkkeihin, mutta sillä oli vaikeuksia ratkaista joitakin suurempia ja vaikeampia tehtäviä. A3 menetelmässä on muutamia parametreja joita tarvitsee säätää tehtäväkohtaisesti, mutta joillakin parametrien arvoilla saadaan kuitenkin hyviä tuloksia kautta linjan. A3 ratkaisee jotkut vaikeammat ongelmat A2:sta paremmin.

Ssa-testeissä A2:sta verrattiin WSATtiin, GSATtiin ja Davis-Putnam-menetelmään hyvin positiivisin tuloksin, ratkaisut löytyivät nimittäin jo noin kymmenesosassa WSATtia nopeammin ja GSAT oli astetta hitaampi. Davis-Putnam oli täydellisenä ratkaisumenetelmänä selvästi hitain eikä onnistunut kohtuullisessa ajassa ratkaisemaan tehtävistä kuin yhden.

Ii-testeissä A2:sta verrattiin GSATtiin, kokonaislukuoptimointiin ja simuloituun jäähdytykseen. DLM oli taas muita menetelmiä huomattavasti parempi. Se ratkaisi ongelmat vain muutamassa sekunnin kymmenyksessä kun näistä menetelmistä toiseksi paras, eli GSAT ei selvinnyt mistään alle sekunnissa.

Kun eri DLM-menetelmiä verrattiin GSATtiin aim, ii, par ja g ongelmissa havaittiin DLM yleisesti ottaen nopeammaksi menetelmäksi. Vain jotkut graafinväritysongelmat GSAT selvitti nopeammin.

DLM:mää verrattiin myös Grasp-menetelmään monilla eri benchmarkeilla. A2-versio on melkein kaikissa tapauksissa taas parempi. Grasp ratkaisee ainoastaan par8-X luokan ongelmat DLM:mää nopeammin.

Koetuloksia esitetty liitteessä 2, täydelliset koetulokset ja implementaatioiden lähdekoodit ovat saatavilla Benjamin Wahin www-sivuilta osoitteesta http://manip.crhc.uiuc.edu/Wah/program.html

6. Yhteenveto

DLM on epätäydellinen menetelmä lauselogiikan toteutuvuuden tarkastamiseen. DLM käyttää jo pitkään tunnettua Lagrangen kertoimien menetelmän uutta diskreettiä versiota ja SAT-ongelma muunnetaan menetelmää varten melko suoraviivaisesti diskreetiksi minimointitehtäväksi. Vaikeampia tetäviä varten perusmenetelmää täytyy tosin hieman muuttaa.
DLM-menetelmän hyötyjä muihin epätäydellisiin menetelmiin ovat se, että Lagrangen kertoimien avulla menetelmä pääsee etenemään lokaaleista minimeistä ilman satunnaisia uudelleenaloituksia. Näin ratkaisu etenee myös jatkuvalla radalla joten jos globaali optimi sijaitsee lokaalien optimien lähellä niin satunnaiset uudelleenaloitukset eivät vie menetelmää kauemmaksi optimista. DLM on myös useimpia muita epätäydellisiä menetelmiä robustimpi koska sen kyky löytää ratkaisu ei ole niin riippuvainen siitä mistä pisteestä algoritmin suoritus aloitetaan. Menetelmässä on vielä kehittämisen varaa, koska A3-versiossa on parametreja joiden arvojen valinnoille ei ole ainakaan tässä paperissa annettu kovin yksityiskohtaisia perusteluita. Saavutetut hyvät tulokset antavat tosin aihetta uskoa, että tämän kaltaisilla menetelmillä olisi enemmänkin annettavaa lauselogiikan toteutuvuustarkastukselle. Menetelmän nykyinen version on vieläkin kilpailukykyinen muiden menetelmien kanssa ainakin joltain osin. Vuoden 2002 SAT-solverien kilpailussa DLM-menetelmät olivat kaikista osallistuneista solvereista parhaita selvittämään satunnaisesti generoituja benchmark-testejä.

Liite 1 DLM algoritmit

DLM $\mathcal{A}_1$

Set initial x
Set $\lambda = 0$
Set $c = 1$
Set $\vartheta = 10$
while x is not a solution, *i.e.*, $N(x) > 0$
  if number of unsatisfied clauses $\leq \vartheta$, **then**
    Maintain a list of unsatisfied clauses
    if $\exists$ variable v in one of the unsatisfied clauses such that
      $L(x', \lambda) < L(x, \lambda)$ when flipping v in x to get x' **then**
      $x \longleftarrow x'$
    else
      Update λ: $\lambda \longleftarrow \lambda + c \cdot U(x)$
    end_if
  else
    if $\exists$ variable v such that $L(x', \lambda) < L(x, \lambda)$ when flipping v
      in a predefined order in x to get x' **then**
      $x \longleftarrow x'$
    else
      Update λ: $\lambda \longleftarrow \lambda + c \cdot U(x)$
    end_if
  end_if
end_while

DLM $\mathcal{A}_2$

Set initial x
Set $\lambda = 0$
Set $c = 1$
Set $\kappa = n/3$, where n is the number of variables
while x is not a solution, *i.e.*, $N(x) > 0$
  if number of iterations $\geq \kappa$ **then**
    Maintain a list, l, of variables such that
      if one of them is flipped, the solution will improve.
    if l is not empty **then**
      Update x by flipping the first element of l
    else
      Update λ: $\lambda \longleftarrow \lambda + c \cdot U(x)$
    end_if
  else
    if $\exists$ variable v such that $L(x', \lambda) < L(x, \lambda)$ when flipping v
      in a predefined order in x to get x' **then**
      $x \longleftarrow x'$
    else
      Update λ: $\lambda \longleftarrow \lambda + c \cdot U(x)$
    end_if
  end_if
end_while

DLM $\mathcal{A}_3$

Set initial x
Set $\lambda = 0$
Set $\kappa = n/3$, where n is the number of variables
Set tabu length L_t, e.g., 50
Set flat-region limit L_f, e.g., 50
Set λ reset interval I_λ, e.g., 10000
Set constant c, e.g., 1/2
Set constant r, e.g., 1.5
while x is not a solution, *i.e.*, $N(x) > 0$
  if number of iterations $\geq \kappa$ **then**
    Maintain a list, l, of variables such that
      if one of them is flipped, the solution will improve.
  end_if
  if number of iterations $\geq \kappa$ **and** l is not empty **then**
    Update x by flipping the first element of l
  else if $\exists$ variable v such that $L(x', \lambda) < L(x, \lambda)$ when flipping v
      in a predefined order in x to get x' **then**
    $x \longleftarrow x'$
  else if $\exists$ v such that $L(x', \lambda) = L(x, \lambda)$ when flipping v in x to get x'
      and number of consecutive flat moves $\leq L_f$
      and v has not been flipped in the last L_t iterations **then**
    $x \longleftarrow x'$ /* flat move */
  else
    Update λ: $\lambda \longleftarrow \lambda + c \cdot U(x)$,
  end_if
  if iteration index *mod* $I_\lambda = 0$ **then**
    Reduce λ for all clauses, e.g. $\lambda \longleftarrow \lambda / r$
  end_if
end_while

Liite 2 Koetulokset

Table 1. Execution times of $\mathcal{A}_2$ in CPU seconds on a Sun SparcStation 10/51 for one run of $\mathcal{A}_2$ starting from x _ 0 (origin) as the initial point on some DIMACS benchmark problems.

Problem Identifier	DLM $\mathcal{A}_2$ Time	DLM $\mathcal{A}_2$ # of Iter.	Problem Identifier	DLM $\mathcal{A}_2$ Time	DLM $\mathcal{A}_2$ # of Iter.
ssa7552-038	0.200	4126	ii16a1	2.100	756
ssa7552-158	0.167	3279	ii16b1	2.100	1313
ssa7552-159	0.167	3802	ii16c1	1.217	916
ssa7552-160	0.167	3409	ii16d1	1.067	595
aim-100-2_0-yes1-1	0.033	982	ii16e1	1.317	1224
aim-100-2_0-yes1-2	0.067	1680	ii32b3	0.883	529
aim-100-2_0-yes1-3	0.000	513	ii32c3	0.533	735
aim-100-2_0-yes1-4	0.367	8510	ii32d3	3.817	1589
par8-2-c	0.317	7841	ii32e3	0.650	518
par8-4-c	0.233	5784			

Table 2. Comparison of $\mathcal{A}_2$'s execution times in seconds averaged over 10 runs with respect to published results of WSAT, GSAT, and Davis-Putnam's algorithm [46] on some of the circuit diagnosis problems in the DIMACS archive. ($\mathcal{A}_2$: Sun SparcStation 10/51 and a 150-MHz SGI Challenge with MIPS R4400; GSAT, WSAT and DP: SGI Challenge with a 70 MHz MIPS R4400.)

Problem Identifier	No. of Var.	No. of Clauses	DLM $\mathcal{A}_2$ SUN	DLM $\mathcal{A}_2$ SGI	DLM $\mathcal{A}_2$ # Iter.	WSAT SGI	GSAT SGI	DP SGI
ssa7552-038	1501	3575	0.228	0.235	7970	2.3	129	7
ssa7552-158	1363	3034	0.088	0.102	2169	2	90	*
ssa7552-159	1363	3032	0.085	0.118	2154	0.8	14	*
ssa7552-160	1391	3126	0.097	0.113	3116	1.5	18	*

Table 3. Comparison of $\mathcal{A}_2$'s execution times in seconds averaged over 10 runs with published results on circuit synthesis problems from the DIMACS archive, including the best known results obtained by GSAT, integer programming (IP), and simulated annealing [46]. ($\mathcal{A}_2$: Sun SparcStation 10/51 and a 150-MHz SGI Challenge with MIPS R4400; GSAT and SA: SGI Challenge with a 70 MHz MIPS R4400; Integer Programming: VAX 8700.)

Problem Identifier	No. of Var.	No. of Clauses	DLM $\mathcal{A}_2$ SUN	DLM $\mathcal{A}_2$ SGI	DLM $\mathcal{A}_2$ # Iter.	GSAT SGI	IP Vax	SA SGI
ii16a1	1650	19368	0.122	0.128	819	2	2039	12
ii16b1	1728	24792	0.265	0.310	1546	12	78	11
ii16c1	1580	16467	0.163	0.173	797	1	758	5
ii16d1	1230	15901	0.188	0.233	908	3	1547	4
ii16e1	1245	14766	0.297	0.302	861	1	2156	3

Table 4. Comparison of DLM's execution times in seconds averaged over 10 runs with the best known results obtained by GSAT [43] on the DIMACS circuit-synthesis, parity-learning, artificially generated 3-SAT instances, and graph coloring problems. Results on $\mathcal{A}_3$ were based on a tabu length of 50, flat region limit of 50, λ reset interval of 10,000, and λ reset to be $\lambda/1.5$ when the λ reset interval is reached. For "*g125-18*" and "*g250-15*," $c = 1/2$; For "*g125-17*" and "*g250-29*," $c = 1/16$. ($\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$: Sun SparcStation 10/51; GSAT: SGI Challenge (model unknown))

Problem Identifier	No. of Var.	No. of Clauses	SUN Sec.	Success Ratio	SGI Sec.	Success Ratio
			DLM $\mathcal{A}_1$		GSAT	
aim-100-2_0-yes1-1	100	200	0.19	10/10	1.96	9/10
aim-100-2_0-yes1-2	100	200	0.65	10/10	1.6	10/10
aim-100-2_0-yes1-3	100	200	0.19	10/10	1.09	10/10
aim-100-2_0-yes1-4	100	200	0.10	10/10	1.54	10/10
			DLM $\mathcal{A}_2$		GSAT	
ii32b3	348	5734	0.31	10/10	0.6	10/10
ii32c3	279	3272	0.12	10/10	0.27	10/10
ii32d3	824	19478	1.05	10/10	2.24	10/10
ii32e3	330	5020	0.16	10/10	0.49	10/10
par8-2-c	68	270	0.06	10/10	1.33	10/10
par8-4-c	67	266	0.09	10/10	0.2	10/10
			DLM $\mathcal{A}_3$		GSAT	
g125.17	2125	66272	1390.32	10/10	264.07	7/10
g125.18	2250	70163	3.197	10/10	1.9	10/10
g250.15	3750	233965	2.798	10/10	4.41	10/10
g250.29	7250	454622	1219.56	9/10	1219.88	9/10

Table 5. Comparison of DLM's execution time in seconds over 10 runs with published results of Grasp on the "aim" problems from the DIMACS archive [39]. (DLM $\mathcal{A}_1$, $\mathcal{A}_2$: Sun SparcStation 10/51; Grasp: SGI Challenge with a 150 MHz MIPS R4400; Success ratio for Grasp is always 10/10.)

Problem Identifier	DLM $\mathcal{A}_2$ Succ. Ratio	DLM $\mathcal{A}_2$ Sun CPU Seconds Avg.	Min.	Max.	DLM $\mathcal{A}_1$ Succ. Ratio	DLM $\mathcal{A}_1$ Sun CPU Seconds Avg.	Min.	Max.	Grasp Avg. SGI Seconds
aim-50-1_6-yes1-1	10/10	0.032	0.000	0.150	10/10	0.042	0.017	0.100	1.12
aim-50-1_6-yes1-2	6/10	0.008	0.000	0.017	10/10	0.020	0.000	0.050	0.12
aim-50-1_6-yes1-3	10/10	0.010	0.000	0.017	10/10	0.027	0.000	0.050	0.49
aim-50-1_6-yes1-4	5/10	0.210	0.017	0.917	10/10	0.030	0.000	0.067	0.46
aim-50-2_0-yes1-1	10/10	0.012	0.000	0.033	10/10	0.035	0.017	0.083	3.67
aim-50-2_0-yes1-2	8/10	0.010	0.000	0.017	10/10	0.035	0.017	0.100	2.82
aim-50-2_0-yes1-3	10/10	0.007	0.000	0.017	10/10	0.077	0.000	0.533	1.76
aim-50-2_0-yes1-4	10/10	0.048	0.000	0.250	10/10	12.603	0.000	125.733	0.62
aim-50-3_4-yes1-1	10/10	0.025	0.000	0.067	10/10	1.528	0.017	13.400	0.39
aim-50-3_4-yes1-2	10/10	0.015	0.000	0.033	10/10	0.162	0.017	0.350	0.58
aim-50-3_4-yes1-3	10/10	0.012	0.000	0.033	10/10	0.142	0.033	0.317	0.49
aim-50-3_4-yes1-4	9/10	0.013	0.000	0.033	10/10	0.057	0.033	0.150	0.91
aim-50-6_0-yes1-1	10/10	0.010	0.000	0.033	10/10	0.027	0.000	0.067	0.06
aim-50-6_0-yes1-2	10/10	0.007	0.000	0.017	10/10	0.028	0.017	0.050	0.14
aim-50-6_0-yes1-3	10/10	0.007	0.000	0.017	10/10	0.035	0.000	0.100	0.05
aim-50-6_0-yes1-4	10/10	0.007	0.000	0.017	10/10	0.027	0.000	0.067	0.05
aim-100-1_6-yes1-1	10/10	0.068	0.033	0.117	10/10	0.092	0.033	0.200	320.40
aim-100-1_6-yes1-2	10/10	0.053	0.017	0.100	10/10	0.098	0.050	0.167	122.27
aim-100-1_6-yes1-3	10/10	0.095	0.050	0.333	10/10	0.142	0.033	0.333	157.15
aim-100-1_6-yes1-4	10/10	0.052	0.000	0.117	10/10	0.095	0.017	0.317	50.10
aim-100-2_0-yes1-1	9/10	0.854	0.017	2.700	10/10	0.193	0.050	0.383	391.79
aim-100-2_0-yes1-2	10/10	0.287	0.050	0.900	10/10	0.652	0.117	1.650	208.40
aim-100-2_0-yes1-3	10/10	0.100	0.017	0.333	10/10	0.187	0.067	0.400	118.22
aim-100-2_0-yes1-4	10/10	0.357	0.033	1.917	10/10	0.097	0.050	0.167	1352.38
aim-100-3_4-yes1-1	10/10	0.450	0.000	2.950	10/10	0.795	0.133	4.417	10.27
aim-100-3_4-yes1-2	10/10	0.195	0.017	1.383	10/10	0.362	0.133	0.817	34.71
aim-100-3_4-yes1-3	10/10	0.050	0.017	0.100	10/10	0.858	0.067	3.800	49.69
aim-100-3_4-yes1-4	10/10	0.038	0.000	0.100	10/10	0.170	0.000	0.317	24.62
aim-100-6_0-yes1-1	10/10	0.020	0.000	0.033	10/10	0.075	0.017	0.133	0.52
aim-100-6_0-yes1-2	10/10	0.018	0.000	0.033	10/10	0.142	0.033	0.467	0.77
aim-100-6_0-yes1-3	10/10	0.017	0.000	0.050	10/10	0.082	0.017	0.233	0.38
aim-100-6_0-yes1-4	10/10	0.018	0.000	0.033	10/10	0.093	0.017	0.267	0.84
aim-200-1_6-yes1-1	10/10	0.958	0.150	2.433	10/10	0.748	0.333	1.583	
aim-200-1_6-yes1-2	6/10	0.786	0.417	1.283	10/10	0.635	0.150	2.350	
aim-200-1_6-yes1-3	10/10	5.498	0.267	43.467	10/10	1.217	0.233	6.950	
aim-200-1_6-yes1-4	2/10	70.042	6.300	133.783	10/10	2.308	0.333	7.183	
aim-200-2_0-yes1-1	1/10	1.283	1.283	1.283	10/10	8.128	0.300	51.483	
aim-200-2_0-yes1-2	4/10	0.538	0.150	1.350	10/10	6.132	0.317	30.000	
aim-200-2_0-yes1-3	7/10	6.355	0.150	28.450	10/10	9.545	0.383	45.717	

Table 5. Continued.

Problem Identifier	DLM $\mathcal{A}_2$ Succ. Ratio	DLM $\mathcal{A}_2$ Sun CPU Seconds Avg.	Min.	Max.	DLM $\mathcal{A}_1$ Succ. Ratio	DLM $\mathcal{A}_1$ Sun CPU Seconds Avg.	Min.	Max.	Grasp Avg. SGI Seconds
aim-200-2_0-yes1-4	0/10				10/10	2.102	0.300	6.950	
aim-200-3_4-yes1-1	8/10	0.469	0.183	1.100	10/10	6.638	0.517	16.500	
aim-200-3_4-yes1-2	10/10	0.547	0.050	3.417	10/10	29.117	0.917	213.467	
aim-200-3_4-yes1-3	10/10	0.838	0.050	5.500	10/10	2.405	0.550	9.467	
aim-200-3_4-yes1-4	9/10	3.122	0.050	22.433	9/10	6.520	0.917	27.150	
aim-200-6_0-yes1-1	10/10	0.075	0.033	0.133	10/10	0.575	0.100	1.717	91.87
aim-200-6_0-yes1-2	9/10	0.209	0.050	0.583	10/10	0.350	0.117	0.933	108.21
aim-200-6_0-yes1-3	10/10	0.102	0.017	0.317	10/10	0.513	0.150	1.250	162.14
aim-200-6_0-yes1-4	10/10	0.218	0.017	0.717	10/10	0.415	0.050	1.000	134.01

Table 9. Execution times of DLM $\mathcal{A}_2$ in Sun SparcStation 10/51 CPU seconds over 10 runs on the "as" and "tm" problems from the DIMACS archive.

Problem Identifier	Success Ratio	Sun CPU Seconds Avg.	Min.	Max.	Problem Identifier	Success Ratio	Sun CPU Seconds Avg.	Min.	Max.
as2	10/10	0.020	0.017	0.033	as10	10/10	0.103	0.067	0.150
as3	10/10	0.037	0.017	0.050	as11	10/10	0.047	0.017	0.067
as4	10/10	0.157	0.133	0.200	as12	10/10	0.038	0.017	0.067
as5	10/10	0.988	0.850	1.283	as13	10/10	0.085	0.067	0.117
as6	10/10	0.098	0.050	0.150	as14	10/10	0.017	0.017	0.033
as7	10/10	0.520	0.450	0.633	as15	10/10	0.157	0.117	0.200
as8	10/10	0.047	0.017	0.067					
tm1	10/10	0.238	0.217	0.250	tm2	10/10	0.013	0.000	0.033

Table 6. Comparison of DLM $\mathcal{A}_2$'s execution time in seconds over 10 runs with published results of Grasp on the "ii" problems from the DIMACS archive [39]. (DLM $\mathcal{A}_2$: Sun SparcStation 10/51; Grasp: SGI Challenge with a 150 MHz MIPS R4400, and success ratio for Grasp is always 10/10.)

Problem Identifier	Success Ratio	DLM $\mathcal{A}_2$ Sun CPU Seconds Avg.	Min.	Max.	Grasp Avg. SGI Seconds
ii8a1	10/10	0.003	0.000	0.017	0.00
ii8a2	10/10	0.007	0.000	0.017	0.03
ii8a3	10/10	0.013	0.000	0.017	0.10
ii8a4	10/10	0.027	0.017	0.033	0.23
ii8b1	10/10	0.012	0.000	0.017	0.11
ii8b2	10/10	0.028	0.017	0.050	1.67
ii8b3	10/10	0.043	0.017	0.067	35.02
ii8b4	10/10	0.062	0.050	0.083	369.37
ii8c1	10/10	0.013	0.000	0.017	37.26
ii8c2	10/10	0.040	0.033	0.050	8.69
ii8d1	10/10	0.018	0.017	0.033	1.19
ii8d2	10/10	0.043	0.033	0.050	3.23
ii8e1	10/10	0.020	0.017	0.033	1.44
ii8e2	10/10	0.040	0.033	0.067	21.97
ii16a1	10/10	0.122	0.117	0.133	23.46
ii16a2	10/10	0.302	0.200	0.433	1970.58
ii16b1	10/10	0.265	0.217	0.350	449.99
ii16b2	10/10	0.377	0.183	0.717	58.41
ii16c1	10/10	0.163	0.133	0.200	20.83
ii16c2	10/10	0.667	0.133	1.350	43.30
ii16d1	10/10	0.188	0.167	0.217	36.09
ii16d2	10/10	0.618	0.250	1.333	56.32
ii16e1	10/10	0.297	0.267	0.367	74.62
ii16e2	10/10	1.273	0.183	3.350	28.06
ii32a1	10/10	0.337	0.133	1.000	68.36
ii32b1	10/10	0.028	0.017	0.033	0.98
ii32b2	10/10	0.130	0.050	0.517	8.08
ii32b3	10/10	0.305	0.150	0.767	8.21
ii32b4	10/10	0.460	0.167	1.033	28.21
ii32c1	10/10	0.022	0.000	0.033	1.79
ii32c2	10/10	0.050	0.033	0.083	0.41
ii32c3	10/10	0.118	0.083	0.233	2.01
ii32c4	10/10	2.940	0.567	6.217	200.97
ii32d1	10/10	0.065	0.017	0.167	2.04
ii32d2	10/10	0.202	0.083	0.833	17.92
ii32d3	10/10	1.047	0.333	2.750	666.73
ii32e1	10/10	0.022	0.017	0.033	17.67
ii32e2	10/10	0.097	0.050	0.183	1.89
ii32e3	10/10	0.160	0.100	0.450	6.04
ii32e4	10/10	0.190	0.150	0.233	11.53
ii32e5	10/10	0.402	0.250	1.450	16.47

Table 7. Comparison of DLM $\mathcal{A}_2$'s execution time in seconds over 10 runs with published results of Grasp on the "jnh," "par," and "ssa" problems from the DIMACS archive [39]. (DLM $\mathcal{A}_2$: Sun SparcStation 10/51; Grasp: SGI Challenge with a 150 MHz MIPS R4400, and success ratio for Grasp is always 10/10.)

Problem Identifier	Success Ratio	DLM $\mathcal{A}_2$ Sun CPU Seconds Avg.	Min.	Max.	Grasp Avg. SGI Seconds
jnh1	10/10	0.068	0.017	0.150	11.87
jnh7	10/10	0.043	0.017	0.083	3.61
jnh12	10/10	0.155	0.067	0.250	0.84
jnh17	10/10	0.082	0.033	0.167	1.66
jnh201	10/10	0.028	0.017	0.050	1.48
jnh204	10/10	0.172	0.017	0.667	14.64
jnh205	10/10	0.103	0.050	0.183	6.17
jnh207	10/10	0.337	0.050	1.817	3.61
jnh209	10/10	0.433	0.067	2.000	7.45
jnh210	10/10	0.033	0.017	0.067	2.35
jnh212	10/10	5.442	0.033	51.250	70.92
jnh213	10/10	0.083	0.017	0.183	9.43
jnh217	10/10	0.060	0.000	0.133	5.76
jnh218	10/10	0.063	0.017	0.183	1.45
jnh220	10/10	0.387	0.033	2.033	10.17
jnh301	10/10	0.333	0.117	0.950	46.23
par8-1	6/10	3.311	0.033	13.117	0.59
par8-2	6/10	3.981	0.400	7.667	1.78
par8-3	4/10	6.012	0.317	18.017	2.69
par8-4	5/10	6.440	1.083	11.017	0.57
par8-5	6/10	11.478	5.850	18.633	0.86
par8-1-c	10/10	0.075	0.000	0.400	0.07
par8-2-c	10/10	0.058	0.000	0.267	0.14
par8-3-c	10/10	1.998	0.000	9.233	0.35
par8-4-c	10/10	0.088	0.017	0.367	0.55
par8-5-c	10/10	0.477	0.017	2.633	0.17
ssa7552-038	10/10	0.228	0.083	0.933	7.05
ssa7552-158	10/10	0.088	0.050	0.167	3.42
ssa7552-159	10/10	0.085	0.067	0.150	1.72
ssa7552-160	10/10	0.097	0.050	0.183	22.13

Table 8. Comparison of DLM $\mathcal{A}_3$'s execution time in seconds over 10 runs with published results of Grasp on some of the more difficult DIMACS benchmark problems from the DIMACS archive [39]. (Success ratio of Grasp is always 10/10.) Program parameters: For all problems, flat region limit = 50; λ reset to $\lambda/1.5$ every 10,000 iterations. For par-16-[1-5] problems: Tabu length = 100, $\lambda \_ 1$. For the rest of par problems:, Tabu length = 50, $\lambda \_ \frac{1}{2}$. For f problems: Tabu length = 50, $\lambda \_ \frac{1}{16}$. For hanoi4 problem: Tabu length = 50, $\lambda \_ \frac{1}{2}$.
System configuration: DLM $\mathcal{A}_3$: Sun SparcStation 10/51; Grasp: SGI Challenge with a 150 MHz MIPS R4400.

Problem Identifier	Succ. Ratio	DLM $\mathcal{A}_3$ Sun CPU Seconds Avg.	Min.	Max.	Grasp Average SGI Sec.
par8-1	10/10	4.780	0.133	14.383	0.59
par8-2	10/10	5.058	0.100	13.067	1.78
par8-3	10/10	9.903	0.350	21.150	2.69
par8-4	10/10	5.842	0.850	16.433	0.57
par8-5	10/10	14.628	1.167	34.900	0.86
par16-1	5/10	11172.8	4630.6	20489.1	9643.38
par16-2	1/10	856.9	856.9	856.9	8993.89
par16-3	1/10	20281.6	20281.6	20281.6	
par16-4	3/10	3523.1	1015.0	7337.9	
par16-5	1/10	13023.4	13023.4	13023.4	
par16-1-c	10/10	398.1	11.7	1011.9	2235.83
par16-2-c	10/10	1324.3	191.0	4232.3	2179.04
par16-3-c	10/10	987.2	139.8	3705.2	2035.23
par16-4-c	10/10	316.7	5.7	692.66	2071.30
par16-5-c	10/10	1584.2	414.5	3313.2	2566.35
hanoi4	1/10	476.5	476.5	476.5	
f600	10/10	16.9	2.1	37.2	
f1000	10/10	126.8	4.4	280.7	
f2000	10/10	1808.6	174.3	8244.7	